Weighted bee colony algorithm for discrete optimization problems with application to feature selection

Moayedikia, Alireza; Jensen, Richard; Wiil, Uffe Kock; Forsati, Rana

Published in:
Engineering Applications of Artificial Intelligence

DOI:
10.1016/j.engappai.2015.06.003

Publication date:
2015

Citation for published version (APA):
Moayedikia, A., Jensen, R., Wiil, U. K., & Forsati, R. (2015). Weighted bee colony algorithm for discrete optimization problems with application to feature selection. *Engineering Applications of Artificial Intelligence*, *44*, 153-167. https://doi.org/10.1016/j.engappai.2015.06.003

Weighted bee colony algorithm for discrete optimization problems with application to feature selection

Alireza Moayedikia[a, *], Richard Jensen[b], Uffe Kock Wiil[c], Rana Forsati[d]

[a] Department of Information Systems and Business Analytics, Deakin University, Victoria, Australia.
[b] Department of Computer Science, Aberystwyth University, Wales, U.K.
[c] The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Odense, Denmark.
[d] Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA.

Abstract

The conventional bee colony optimization (BCO) algorithm, one of the recent swarm intelligence (SI) methods, is good at exploration whilst being weak at exploitation. In order to improve the exploitation power of BCO, in this paper we introduce a novel algorithm, dubbed as weighted BCO (*w*BCO), that allows the bees to search in the solution space deliberately while considering policies to share the attained information about the food sources heuristically. For this purpose, *w*BCO considers global and local weights for each food source, where the former is the rate of popularity of a given food source in the swarm and the latter is the relevancy of a food source to a category label. To preserve diversity in the population, we embedded new policies in the recruiter selection stage to ensure that uncommitted bees follow the most similar committed ones. Thus, the local food source weighting and recruiter selection strategies make the algorithm suitable for discrete optimization problems. To demonstrate the utility of *w*BCO, the feature selection (FS) problem is modeled as a discrete optimization task, and has been tackled by the proposed algorithm. The performance of *w*BCO and its effectiveness in dealing with feature selection problem are empirically evaluated on several standard benchmark optimization functions and datasets and compared to the state-of-the-art methods, exhibiting the superiority of *w*BCO over the competitor approaches.

Keywords: Bee Colony Optimization; Categorical Optimization; Classification; Feature Selection; Weighted Bee Colony Optimization.

1. Introduction

Swarm intelligence (SI) is one of the well-known classes of optimization and refers to algorithms relying on the intelligence of a swarm to locate the best parts of the solution space. Particle swarm optimization (PSO) [**1**], ant colony optimization (ACO) [**2**] and BCO [**3**] [**4**], are examples of SI algorithms. Many problems such as text clustering [**5**], feature selection [**6**] [**7**] [**8**], etc., can be modeled as discrete optimization problems and solutions obtained through SI algorithms.

*Corresponding author: Alireza Moayedikia, is a PhD student in Information Systems, Department of Information Systems and Business Analytics, Faculty of Business and Law, Deakin University 70 Elgar Road, Burwood, 3125, Victoria, Australia.
amoayedi@deakin.edu.au

BCO is one of the most recent developments of swarm intelligence proposed by Teodorovic and colleagues [**4**], which has been successfully applied to many fields of science including image analysis [**9**], bioinformatics [**10**], etc. The algorithm simulates the natural behavior of the bees in locating food resources. In summary, the BCO algorithm has five main stages: 1) initialization, 2) solution creation, 3) fitness assessment, 4) loyalty measurement, and 5) recruiters selection.

In the first step, the algorithm parameters are initialized (*initialization*). Then in the second step the solutions are created, partially in the sense that the whole solution will not be created at once while during several forward and backward steps a complete solution will be created (*solution creation*). In BCO a forward step occurs once the bees leave their hive to create solutions and explore the solution space, while the backward stage occurs once the bees return to their hive to measure the goodness of the produced solutions, share the attained information and finally select the follower and recruiters.

During the solution creation steps, after each forward movement, the bees return to their hive to assess the solutions (*fitness assessment*). The fitness assessment occurs in the backward step, where each bee also measures how loyal it is to the created partial solution (*loyalty measurement*). Finally, before performing the next forward movement, the bees must be divided into two sets of committed (recruiter) and uncommitted (followers) bees to decide which bees will follow the other bees (*recruiter selection*). Within a generation the algorithm iterates between the second and the fifth stages until all the bees create their full solutions. For further details about BCO interested readers may refer to the work of Forsati and colleagues [**11**].

The advantage of BCO is its ability in tuning the search direction in the early stages of exploration, while other SI algorithms such as ACO require full traversals by all ants to adjust pheromone weights accurately and finally identifying the worthwhile exploration paths. Once the bees perform the backward movement they in fact try to distinguish worthwhile and non-worthwhile solution paths. This action allows the search direction to be tuned toward the most optimal parts of the solution space found so far. Similarly, PSO has the same characteristic, in which the flock of birds flies toward the global and/or local best solutions while exploring the solution space.

Mainly swarm intelligence algorithms (including BCO) rely on randomness to search the solution space. This might give absolute freedom to the swarm to search the solution space, but randomness might degrade the exploitation ability of BCO, in the sense that the worthwhile parts of the space remain undiscovered or unintentionally ignored. In other words, the bee colony has weak exploitation ability while having a good level of exploration [**12**]. Therefore to increase the exploitation ability of BCO, we introduce a new variation called weighted bee colony optimization (*w*BCO) which considers new policies in measuring the loyalty degrees of the bees and also recruiter selection. The formulation of recruiter selection will make the algorithm applicable for classification and regression problems.

As explained, each backward step has three stages: fitness assessment, loyalty assessment, and recruiter selection. In the backward step for *w*BCO, where the bees measure how loyal they are to their created (partial) solutions, the algorithm considers two weights for each food source. One is a global weight, which measures how popular a given food source is in the swarm and the other is a local weight, which indicates the extent to which a selected food source can contribute to the category label of the classification problem. In the recruiter selection

step, in order to preserve diversity the followers select their recruiters in a filtering stepwise process. We apply two filtering stages; one is based on similarity, and the other based on fitness values. In similarity filtering, for a given follower a set of recruiters is selected based on the traversal similarity and then the follower selects a recruiter bee which has the closest fitness value. This recruiter selection strategy is only applicable if the variables of a classification problem only accept discrete values (e.g., integers, binary, letters).

To investigate the effectiveness of the proposed algorithm in application, we further applied the proposed *w*BCO to feature selection (FS) and modeled the curse of dimensionality as a discrete optimization task to investigate if *w*BCO can have applicability in classification tasks. Also, other applications such as text classification can be modeled using *w*BCO, but as a result of wide applications of FS including bioinformatics [**13**], systems monitoring [**14**], text mining [**15**], image processing [**16**], etc., we decided to model FS as a discrete optimization task with *w*BCO. The new feature selection algorithm is called FS-*w*BCO and successful implementation of FS-*w*BCO will indicate that the proposed *w*BCO is also applicable in the fields relying on FS. The contributions of this paper are summarized as follows.

- Modifying the loyalty assessment of the original BCO with the aim of using heuristics to weight the worth of each selected food source and consequently improving the exploitation power of BCO. For this purpose we use the global and local weight of a selected food source.
- In the introduced weighting scheme, each selected food source has two weights: local and global. In the former the algorithm measures how popular the food source is in the swarm, while in the latter the algorithm determines the extent to which the selected food source is relevant to a category label.
- In line with exploitation improvements, we modify the recruiter selection of the original BCO with the aim of using heuristics to preserve diversity in the bees' population by assigning each uncommitted bee to the most similar committed one.
- To investigate the utility of *w*BCO, feature selection is modeled as a discrete optimization problem resulting in another algorithm known as FS-*w*BCO. Experiments are carried out to investigate the efficacy of both *w*BCO and FS-*w*BCO.

The rest of the paper is organized as follows: Section 2 briefly reviews some of the recent literature in the area of bee colony improvements. In Section 3, the new bee colony optimization algorithm, *w*BCO, is proposed. In Section 4, the application of *w*BCO for feature selection is introduced. Section 5 provides some experimentation to show the effectiveness of *w*BCO and the feature selection algorithm (FS-*w*BCO) and finally Section 6 concludes the paper and lists future work.

2. Literature review

As we are introducing a new BCO algorithm, the focus of this section is on some of the recent developments of bee colony-based algorithms. Regarding feature selection algorithms, interested readers can refer to [**17**]. In the literature, two approaches to bee colony-based algorithms are proposed. One is the artificial bee colony (ABC) algorithm proposed by Karaboga and colleagues [**18**] [**19**] and the other is BCO proposed by Teodorovic and colleagues [**4**]. As both of the algorithms rely on the natural behavior of the bees, we also consider the ABC algorithm in this paper.

As shown in Table 1, bee colony improvements mainly aim at improving either the exploration or the exploitation of the algorithm. Hence in this review, we divide the bee colony improvements into these two categories. A third category is related to algorithms targeting improvements in both exploration and exploitation powers of BCO.

Table 1 – An overview of the reviewed articles

Articles	Exploration	Exploitation	Integrated (exploration and exploitation)
Kumar and colleagues [**20**]	√		
Lu and colleagues [**21**]			√
Huang and Lin [**22**]	√		
Kumar [**23**]			√
Forsati and colleagues [**11**]			√
Alzaqebah and Abdullah [**24**]		√	
Karaboga and Akay [**25**]			√
Gao and Liu [**26**]	√		
Li and colleagues [**27**]		√	
Akbari and colleagues [**28**]			√
Imanian and colleagues [**29**]		√	
Alatas [**30**]		√	
Kashan and colleagues [**31**]			√
*w*BCO		√	

There are some BCO algorithms focusing on improvements of exploration power. Gao and Liu [**26**] proposed IABC that uses differential evolution, which is suitable for global optimization. The paper proposes two different variations namely *ABC/rand/1* and *ABC/best/1*.In order to benefit from the advantages of both variations and to reduce the shortcomings, the authors propose a selective probability p to obtain a new search mechanism. In addition, to enhance the global convergence speed, when producing the initial population, both the chaotic systems and the opposition-based learning method are used.

Open shop scheduling (OSSP) is one of the most time-consuming tasks in scheduling problems which can also benefit from BCO algorithms. The range of the solution space is technically downsized by many artificial intelligence algorithms but in most scheduling algorithms every partial solution still needs to be completely solved before this solution can be evaluated. In order to tackle this, Huang and Lin [**22**] propose a new bee colony optimization algorithm, with an idle-time-based filtering scheme, according to the inference of "the smaller the idle-time, the smaller the partial solution", and "the smaller the make span will be". It can automatically stop searching for a partial solution with insufficient profitability while the

scheduler is creating a new scheduling solution, and consequently save time–cost for the remaining partial solution.

Forsati and colleagues [**11**] introduce a new bee colony algorithm called improved BCO (IBCO) and applied it to text clustering problem. The algorithm uses two concepts of cloning and fairness to improve exploration and exploitation power of the bees. Through cloning the algorithm uses the information of previous traversals when it is creating a new solution. Fairness gives every bee the chance to be followed. However the algorithm still suffers from entrapment in local optima. To overcome this problem IBCO was integrated with *k*-means, and four different variation introduced.

Also there are other type of algorithms focusing on exploitation improvements. Alzaqebah and Abdullah [**24**] proposed a new BCO based variation and utilize it to solve the examination scheduling problem. The authors think that selection of a recruiter that searches for a food source or a follower based on a roulette wheel is a drawback. Hence they introduce three selection strategies of tournament, rank, and disruptive selection (DBCO) to overcome this defect and preserve diversity in the population.

Li and colleagues [**27**] proposed another algorithm called improved ABC (I-ABC). The algorithm uses the best-so-far solution, inertia weight, and acceleration coefficients to modify the search process. The purpose of the introduction of the inertia weight and acceleration coefficients is to use them as fitness functions. I-ABC is good in finding the global optimum and convergence speed. In order to have a successful application to optimization problems, a population-based optimization algorithm that realizes rapid convergence and high diversity is needed. At the same time, a good population-based optimization algorithm should have a stable performance regardless of initial population selection. In order to achieve these goals and combine the advantages of I-ABC, ABC and the best-so-far solution, the authors introduced a compounding high-efficiency ABC algorithm with the abilities of prediction and selection (PS-ABC).

Imanian and colleagues [**29**] proposed a new velocity based ABC algorithm (VABC), to overcome the weakness of ABC in exploitation, through applying a new search equation in the onlooker phase. The modified strategies are introduced to focus the new candidate solution towards the global best solution. The work is inspired by the search mechanism of PSO, in which a new neighborhood search strategy is proposed for onlookers. The aim of the approach of Imanian and colleagues is to combine the exploration process of ABC and the exploitation strategy of PSO to improve the optimization process. Hence, the authors consider three main steps in their algorithm. In the first step, the employed bees go on to their food sources and evaluate their nectar amounts and then share the nectar information of the sources with onlookers. In the second step, the best solutions explored in the history are used to direct the movement of the population. To this end, the explored solutions will be chosen depending on the probability values associated with the solutions based on their corresponding fitness values. Finally, in the third step, when the food source positions are not replaced continuously over the predefined number of trials limit, employed bees will abandon the food source positions and become scout bees. Another example of chaotic based improvements can be found in the work presented by Alatas [**30**] in which the author proposed seven new chaotic based bee colony algorithms.

There are some algorithms focusing on improvements of BCO exploration and exploitation powers of bee colony. Even though improvements of either exploration or exploitation power seems helpful, but expecting not be as effective as enhancing both exploitation and exploration powers. Therefore to provide further improvements of BCO some researchers aim at improving both of these powers.

Kumar and colleagues [**20**] proposed a multi-objective directed bee colony optimization algorithm (MODBC). The authors claim that the early and classical optimization techniques such as direct search and gradient methods are not able to provide global optimization solutions. Hence MODBC is an integration of deterministic search, a multi-agent system (MAS) environment, and a bee decision-making process. One of the objectives of this hybridization is to obtain a unique and fast solution and hence generate a better Pareto front for multi-objective problems. MODBC is further applied for solving a multi-objective problem of optimizing the conflicting economic dispatch and emission cost with both equality and inequality constraints.

Another area that BCO can be applied to is dynamic economic dispatch (DED). Lu and colleagues [**21**] propose a chaotic differential bee colony optimization algorithm (CDBCO) and utilize it to address the DED problem considering valve-point effects. In CDBCO, chaotic sequences is used in order to generate candidate solutions and a new searching mechanism based on DE/*best*/1 strategy and finally increasing the exploration ability of BCO. Also, a chaotic local search (CLS) method is used to help BCO overcome the drawback of premature convergence and increase the local exploitation capability.

Kumar [**23**] presents a new BCO algorithm based on the Nelder-Mead method. This method relies on four geometric operations of reflection, expansion, contraction, and shrinking. The author considers two different approaches to finally select the best bee of the population. One is consensus and the other is quorum. In the former, inspired by PSO, the global best is selected while in the latter the number of bees selecting a solution higher than a given threshold is used as the representative solution.

Karaboga and Akay [**25**] proposed another variation suitable for constraint-based optimization and for constraint handling. The proposed algorithm has three main modifications. In the unconstrained optimization algorithms of ABC, only one randomly chosen parameter is changed and the other parameters are copied from the previous solutions. Previously, a random number for each parameter is generated and if it is lower than the modification rate the parameter is changed. Also, the algorithm uses Deb's rule, which has three simple heuristic rules along with a probabilistic selection scheme for feasible solutions based on their fitness values and infeasible solutions based on their violation values. The third modification is the consideration of different probabilities for infeasible and feasible solutions. Then using a roulette wheel selection mechanism, the onlookers and employed are assigned to each other.

Akbari and colleagues [**28**] proposed another ABC algorithm called multi-objective artificial bee colony (MOABC). The algorithm uses a grid-based approach to assess the Pareto front maintained in an external archive. The employed bees adjust their flight paths according to the non-dominated solutions preserved in the archive. Also, the onlooker bees select the food sources advertised by the employed bees to update their positions. The MOABC uses the ε dominance method for updating. In the ε-dominance method, a space with dimensions equal to the number of the problem's objectives will be assumed. Each dimension will get sliced in an ε by ε size. This will break the space to boxes like squares, cubes, or hyper-cubes for two, three,

and more than three objectives, respectively. Pareto dominance is used to assess the qualities of the selected food sources. The scout bees are used by the algorithm to eliminate food sources with poor quality.

Kashan and colleagues [**31**] proposed a new bee colony based algorithm for binary optimization. *DisABC* uses a new differential expression, which employs a measure of dissimilarity between binary vectors instead of the vector subtraction operator typically used in the original ABC algorithm. Such an expression helps to maintain the major characteristics of the original and is respondent to the structure of binary optimization problems too. Similar to the original ABC algorithm, the differential expression of *DisABC* works in a continuous space while its consequence is used in a two-phase heuristic to construct a complete solution in a binary space. The effectiveness of the proposed approach was tested on benchmark test problem instances of the incapacitated facility location problem (UFLP), and compared with two binary optimization algorithms, binDE and PSO, where the results demonstrate that their approach is competitive.

3. *w*BCO: Weighted Bee Colony Optimization

In this section, *w*BCO is proposed as an improvement over the conventional BCO algorithm [**4**]. BCO is good at exploration, while weak at exploitation [**12**]. One of the facts that might limit the exploitation power is the reliance on random decision making. Typically, each algorithm only reaches better solutions than other algorithms for some particular problems. Figure 1, provides an overview of the proposed algorithm.

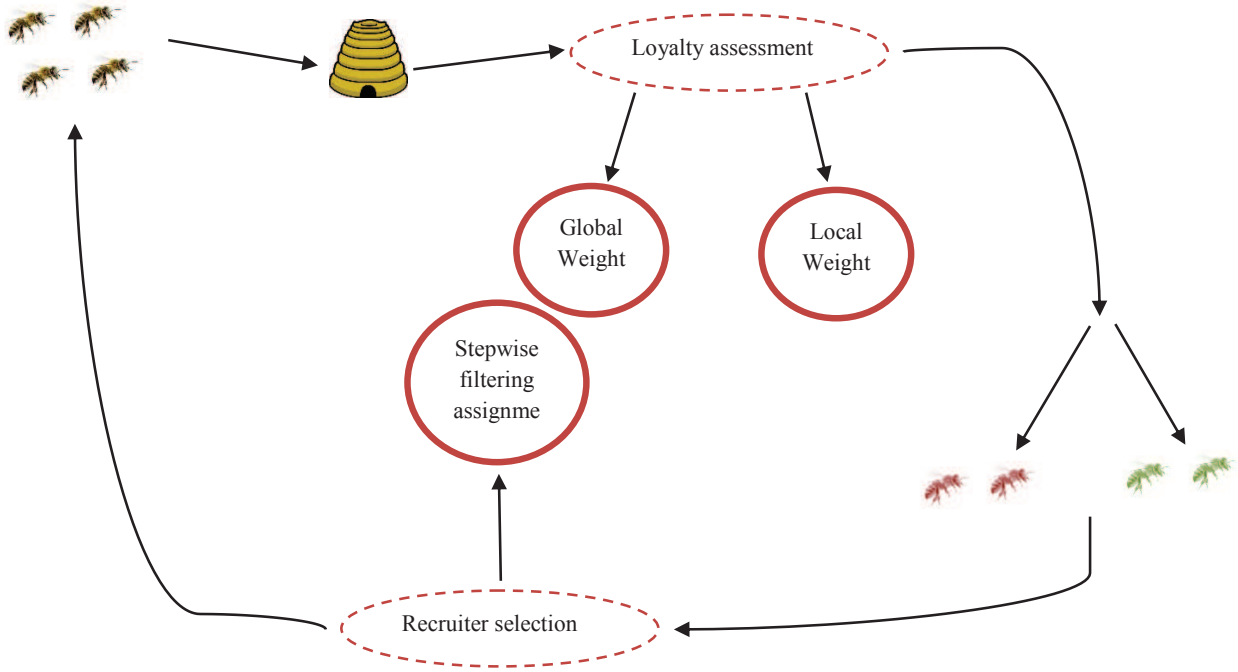

Figure 1 - A block diagram of *w*BCO

In BCO, the loyalty assessment of the bees is dependent only on the fitness values. This kind of assessment provides superficial knowledge about the paths and solutions that the swarm has already created. We believe that through mining the paths traversed by the swarm along with a consideration of fitness values, the loyalty degree of the bees will be judged better. Furthermore, in the conventional BCO, a roulette wheel process will assign the uncommitted bees to the committed ones. This sort of assignment would degrade the performance as some committed solutions might be ignored unintentionally. This case is discussed in more detail later in this section. To tackle these issues, as shown in Figure 1, *w*BCO reduces the reliance of BCO on randomness in the exploitation step and introduces some heuristics for BCO execution to improve the exploitation power. *w*BCO measures the loyalty degree of the bees through weighting procedures (i.e. local and global weighting procedures) and then uses stepwise assignment to identify the recruiters of uncommitted bees.

3.1. Initialization

In this stage the algorithm parameters are initialized, such as the number of bees (*B*), iterations (*G*), and the size of constructive steps (*NC*). The initialization of these parameters can be either random or user-specified. In the *w*BCO algorithm, the number of bees, iterations, and constructive steps are user-specified.

Randomization will cause different initialization scenarios in the sense that different values are generated randomly for different variables. This will prolong the experimentation, since it is required to increase the number of experiments to cover all the possible scenarios. User-specified initialization requires fine-tuning of the variables before running the main algorithm and then executing the algorithm with the most appropriate values.

3.2. Creating partial solutions

As the bees leave their hive, each will take *NC* random steps to create a partial solution. Therefore, the initial partial solution has size *NC*, and grows iteratively in the next forward steps with the maximum size of *NC*. The execution of this stage is similar to the original BCO algorithm and each bee decides randomly to explore the solution space. The pseudo code of the partial solution creation process is shown in Algorithm 1.

As outlined in Algorithm 1, each bee creates its partial solution with the size NC as shown in Lines 1 to 5. The condition of selection of a food source as shown in Line 4 is to check the previously traversed food sources to make sure that the current food source has not been considered in the previous traversals of the same bee. If not, then the bee would consider it as the next possible selection. Once the next food source is selected by all the bees then the length of the solution will increase. This stage terminates once all the bees have created their partial solutions, with the size of NC or lower.

Algorithm 1: Partial solution creation

Input:
An initialized population
NC: the size of constructive step

Output:
A partial solution

Algorithm:

1. *Len* = 0
2. **while** *Len* <*NC* //(i.e. partial solutions with the size *NC* is not created)
3. **foreach** bee *b*
4. Randomly select a food source which has not been selected before by the same bee *b*.
5. **end foreach**
6. *Len* = *Len* +1; //(i.e. increasing the length of currently created solutions by one)
7. **end while**

3.3. Loyalty assessment

Loyalty assessment measures the degree of certainty of a bee to reach optimal parts of the solution space. In *w*BCO this certainty level relies on the fitness value of a bee and the overall weights that a bee gives to the selected food sources. The algorithm weights the worth of a food source from two different perspectives, global and local, as outlined in Equation (1):

$$w(B_{ji}) \propto local_j(i) \times global(i) \tag{1}$$

where B_{ji} is the overall weight of the *i*th food source of the *j*th solution (or of the *j*th bee), $local_j(i)$ identifies the local weight of the *i*th food source in the solution of the *j*th bee and $global(i)$ is the global acceptance of the *i*th food source in the swarm. $global(i)$ is measured according to Equation (2):

$$global(i) = \begin{cases} \frac{\sum_{k=1}^{Sel_i} Fit(k) \times \sum ignored(i)}{\sum_{m=1}^{Unsel_i} Fit(m) \times \sum Select(i)} & Unsel_i \neq 0 \; and \; Sel_i \neq 0 \\ \frac{\sum_{k=1}^{Sel_i} Fit(k)}{\sum Select(i)} & Unsel_i = 0 \; and \; Sel_i \neq 0 \\ 0 & Sel_i = 0 \end{cases} \tag{2}$$

where Sel_i and $Unsel_i$ are the number of bees that have selected and unselected the *i*th food source, respectively. $\sum_{k=0}^{Sel_i} Fit(k)$ is the summation of the fitness value of the bees that have selected the *i*th food source, $\sum ignored(i)$ is the total number of bees that have ignored the *i*th food source, $\sum_{m=0}^{Unsel_i} Fit(m)$ is the summation of the fitness value of the bees that have unselected the *i*th food source, and finally $\sum Select(i)$ is the total number of bees that have selected the *i*th food source.

Algorithm 2 shows the global weight evaluation. In Line 3 if the global weight of a food source has not been computed before by another bee, then it will be evaluated through Lines 4 to 9. This process iterates for all the food sources (Lines 2 to 10) and for all the bees (Lines 1 to 11). In cases where a food source has been selected by all the members of the swarm, $Unsel_i$ is zero, and the global weight of a food source which has not been selected by any of the swarm members is equal to zero.

Algorithm 2: Global weight assessment

Input:
A set of solutions

Output:
The global weight of each selected food source

Algorithm:

1. **foreach** solution S_j
2. **foreach** selected food source f_i in solution S_j
3. **if** the global weight of food source f_i has not been measured before
4. Count the number of bees B_j^S that have food source f_i
5. Count the number of bees B_j^I that ignored food source f_i
6. Sum the fitness of the bees Fit_j^S that have food source f_i
7. Sum the fitness of the bees Fit_j^I that ignored food source f_i
8. Measure the global weight of f_i in S_j according to Equation (2);
9. **end if**
10. **end foreach**
11. **end foreach**

Merely considering the fitness value of the bees that have selected/ignored a food source (Equation (3)) cannot help to decide whether a food source is worthwhile or not. In this regard, consider the following two scenarios:

- Scenario 1: a bee population of size B in which Sel_i and $Unsel_i$ number of bees have selected and ignored the ith food source respectively where $Unsel_i < Sel_i$ and $\sum F_{Unsel_i} > \sum F_{Sel_i}$.
- Scenario 2: a bee population of size B in which a given food source is selected and ignored by Sel_i and $Unsel_i$ number of bees where $Unsel_i > Sel_i$ and $\sum F_{Unsel_i} < \sum F_{Sel_i}$.

$$global(i) = \frac{\sum_{k=1}^{Sel_i} Fit(k)}{\sum_{m=1}^{Unsel_i} Fit(m)} \tag{3}$$

One food source might be selected frequently by the bees having low rates of fitness (scenario 1). On the other hand, a given food source might be ignored by many of the bees in the population while fewer bees with high rates of fitness have selected the same food source (scenario 2). Based on these two scenarios and Equation (3), it will not only be required to consider the fitness value for the global food source weighting but also the frequency of selection/ignorance of that food source should be taken into account.

If the number of bees that have ignored a worthwhile food source is overlooked, then it is likely that the cumulative summation of the fitness of such bees exceed the fitness value of the bees that have selected the same food source. Then in this circumstance, the bees' flying direction will not be changed toward the selection of the most significant food sources, and finally the ignorance of the most significant food sources will lead to a reduced possibility of selection of worthwhile food sources.

In simpler terms, a high frequency of selection of a less significant food source causes the bees to fly toward poor regions of the solution space, which is not desirable. In this paper, this problem is termed "misleading frequency". Hence, according to Equation (4), normalization is required, and is achieved through dividing the cumulative summation of the fitness of the bees that have selected/ignored a given food source by the number of bees that have selected/ignored the same food source. Equation (4) is written in simpler form as Equation (2), where $Unsel_i$ and Sel_i are not zero.

$$global(i) = \frac{\frac{\sum_{k=1}^{Sel_i} Fit(k)}{\sum selected(i)}}{\frac{\sum_{m=1}^{Unsel_i} Fit(m)}{\sum ignored(i)}} \quad (4)$$

Hence true measurement of the global weight of a given food source depends on having the number of bees that have selected/ignored the food source and also the fitness value of the bees which have selected/ignored the same food source. The proposed algorithm is suitable for classification and regression problems as a result of the local weighting policy presented in Equation (5). The local weight identifies the weight of a food source in the *j*th created partial solution through measuring the degree of correlation between a specific food source and a category label. The local weighting is defined as follows:

$$local_j(i) = \frac{\sum_{x=1}^{X} Cor(C_p^x, F_i^j)}{\sum(correlation_i)} \times Fit(B_j) \quad (5)$$

Here, $Fit(B_j)$ is the fitness of the *j*th bee, $\sum_{x=1}^{X} Cor(C_p^x, F_i^j)$ is the correlation (or dependency) between the *i*th traversed food source of the *j*th bee, F_i^j, and the *x*th predicted category label, C_p^x and $\sum(correlation_i)$ is the summation of the correlations (or dependencies) of the traversed food sources to the predicted category label, C_p. For an unselected food source the local weight is zero; x is the total number of predicted category labels. Depending on the problem being formulated through *w*BCO, the number of predicted category labels can vary from one to the total number of possible category labels. Algorithm 3 clarifies the local weighting scheme.

Algorithm 3: Local weight assessment

Input:
A set of solutions

Output:
The local weight of each food source in each solution

Algorithm:

1. **foreach** solution *sol*
2. **foreach** predicted category label C_p^x
3. **foreach** selected food source F_i^j in *sol*
4. Sum the correlations between selected food source F_i^j and the predicted category label C_p^x;
5. **end foreach**
6. **foreach** selected food source F_i^j
7. Calculate the local weight of selected food source F_i^j according to Equation (5);
8. **end foreach**

For each bee (Lines 1 to 8) we measure how dependent the predicted category labels are to the selected food sources (Lines 2 to 5). Then for each food source in a given solution, the algorithms computed the summation of dependency between all the selected food sources in a solution to a predicted category label (Lines 3 and 4). Finally using Equation (5), the algorithm measures the local weight of each selected food source (Lines 6 and 7).

We believe that if the contribution or relevancy of a traversed food source toward a category label is high, then the selected food sources would be salient. In simpler terms, if the traversal of a food source leads a bee towards classifying data more accurately, then the selection of that food source is worthwhile. However, a subset of tightly-correlated food sources to a category label does not necessarily mean assignment of high local weight to the solution.

Assume a set of food sources has led to a poor quality solution while every pair of food source and category label has a high correlation degree. Therefore the local weighting scheme will assign high weights to the solution. This problem is known as "misleading correlation" and is alleviated by considering the fitness value of the solution assessed by the bee. If the fitness value and the average of the local weights in a solution is high, then the local weight in the procedure will consider the solution with high quality, unless otherwise. Therefor it is essential to consider the fitness value to appropriately adjust the local weight of a traversed food source. The final loyalty assessment can be written as Equation (6).

$$LoyaltyDegree(B_j) = Fit(B_j) \times \frac{\sum_{x=1}^{X} Cor(C_p^x, F_i^j)}{\sum(correlation_i)} \times \frac{\sum_{k=1}^{Sel} Fit(k) \times \sum ignored(i)}{\sum_{m=1}^{Unsel} Fit(m) \times \sum Select(i)} \quad (6)$$

Once the loyalty degrees are computed, different strategies for distinguishing loyal (committed) and non-loyal (uncommitted) bees can be considered. The bees can be ranked in descending order based on their loyalty degrees and the first half is considered as committed while the rest is uncommitted. The other strategy is to calculate the average of the loyalty degrees, then bees with loyalty degrees higher than the average are considered as loyal and the rest as non-loyal. Since the algorithm utilizes two weights for each food source to finally measure the loyalty degree of a bee this new variation of BCO is called weighted bee colony optimization (*w*BCO).

3.4. Fitness and performance evaluation

One of the important components of optimization algorithms is to measure the optimality degree of generated solutions. This can be measured through the fitness function. The fitness function of *w*BCO, as a discrete optimization algorithm, depends on the problem being solved by the algorithm. For instance, in feature selection algorithms [**6**] the fitness function would typically be classification accuracy or statistical measures. Similarly in text mining algorithms [**5**] the F-measure, purity, and entropy can measure the degree of optimality of the generated solutions.

3.5. Stepwise filtering recruiter selection

After dividing the bees into two groups, committed (recruiter) and uncommitted (follower), then for each follower, one and only one recruiter should be identified. In the first variation of BCO [**4**], the bees are assigned based on a roulette wheel approach. In this case, an assignment of the bees would result in two different scenarios:

- All or most of the uncommitted bees may follow a small number of committed ones, while a large number of committed bees remain without any followers, or
- It is likely that each committed bee will be followed by at least one uncommitted one.

In the first scenario, as shown in Figure 2 in the grey box, the population is less diverse in the sense that most of the bees have similar solutions which may lead to premature convergence. This type of assignment will also reduce the exploitation power of the bees. In order to alleviate this effect, we propose a stepwise filtering assignment that makes the algorithm suitable for discrete optimization problems. Firstly, an uncommitted bee is more likely to follow a committed one, if both have selected similar food sources during their forward movements. Equation (7) represents a mathematical model of this idea.

$$FollowingProbability_{cb}^{ub} = \frac{\sum_{i=0}^{cs} E_i}{\sum food\_source} \quad (7)$$

where $FollowingProbability_{cb}^{ub}$ is the probability that a committed bee *cb is* to be followed by the uncommitted bee *ub* and $\sum_{i=0}^{cs} E_i$ is the number of commonly-selected food sources by both the uncommitted and committed bees. $\sum food\_source$ is the total number of food sources in the solution space. However, it is likely that one uncommitted bee has the same following probability rate as more than one committed bee. Hence, consideration of the fitness function will enforce the uncommitted bee to select one and only one recruiter, according to Equation (8):

$$FollowingProbability_{cb}^{ub} = \frac{\sum_{i=0}^{ce} E_i}{\sum food\_sources} \times [1 - (F_{Com}^{cb} - F_{uncom}^{ub})] \quad (8)$$

where F_{com}^{cb} is the fitness value of the *cb*th similar committed bee and F_{uncom}^{ub} is the fitness value of the uncommitted bee. Therefore, the lower the differences between fitness values ($F_{Com}^{cb} - F_{uncom}^{ub}$), the higher the value of $[1 - (F_{Com}^{cb} - F_{uncom}^{ub})]$ which increases the probability of following. In problems where the fitness function does not satisfy the conditions $|F_{Com}^{cb}|<1$ and $|F_{uncom}^{ub}|<1$, the fitness values should be normalized within the range [-1, 1]. Also in Equation (8), $F_{uncom}^{ub} \neq F_{Com}^{cb}$ is always true since if $F_{uncom}^{ub} = F_{Com}^{cb}$ then both of the bees are either committed or uncommitted.

Figure 2 shows the difference between random and stepwise filtering assignments of the bees. In Figures 2 and 3 each row corresponds to a bee. The first column is the bee identifier, the second through the sixth columns each corresponds to a food source, where values of 1 and 0 refer to the selection of a food source or otherwise. Binary digits are shown as a symbol of discrete optimization, but letters or integer numbers can also be used. The seventh column indicates if the bee is committed (C) or uncommitted (U), and finally the last column is the fitness value of the bee.

The first advantage of this type of assignment is to help the algorithm to improve the exploitation power through following the most similar bee which also preserves diversity in the population as shown in the white box of Figure 2. In Figure 3, the stepwise filtering assignment is shown. In the first step, committed bees are filtered based on their similarities to the uncommitted ones, and in the second step, similar bees are filtered according to their fitness values.

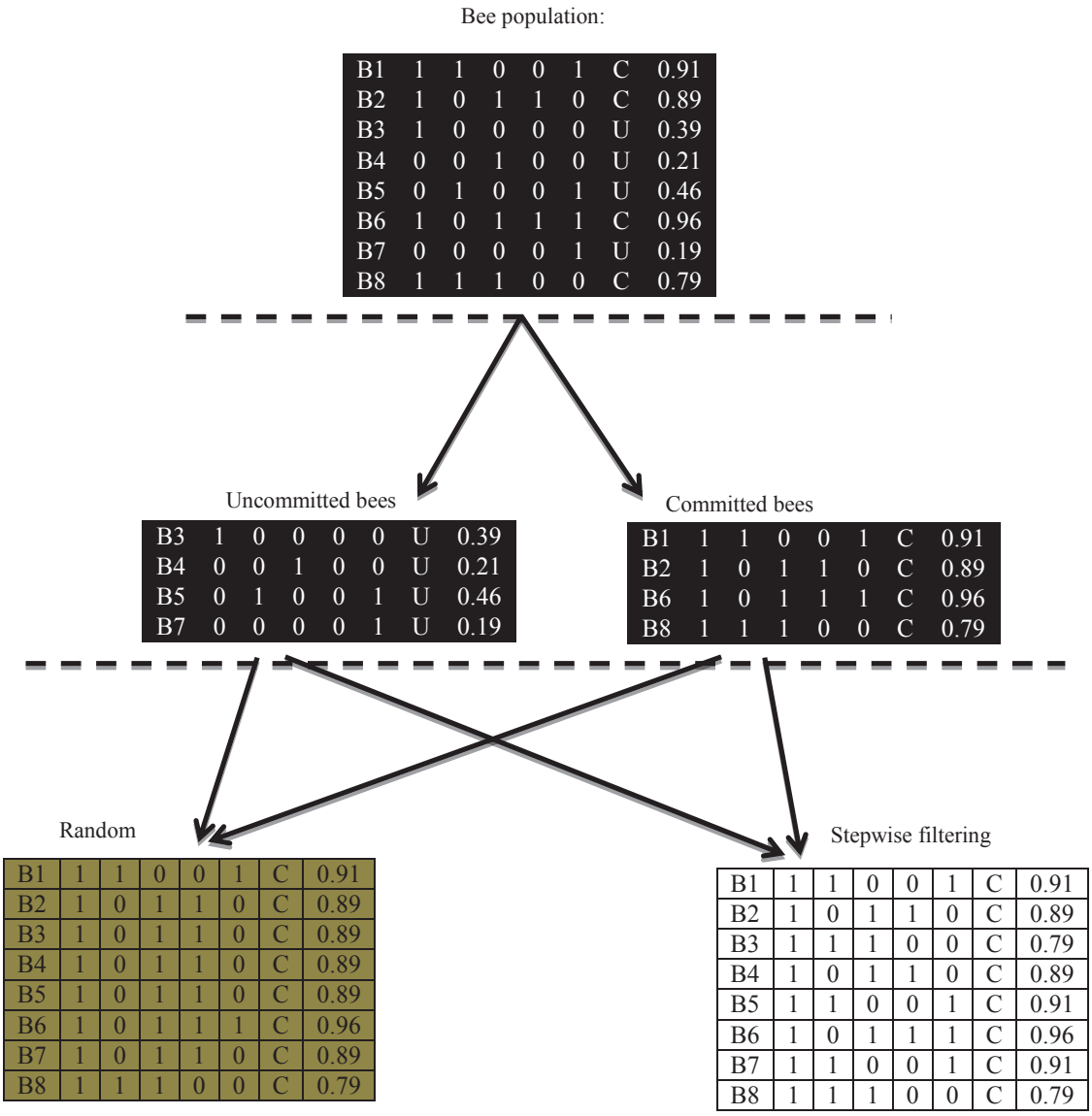

Figure 2 – Stepwise filtering assignment in binary solution space

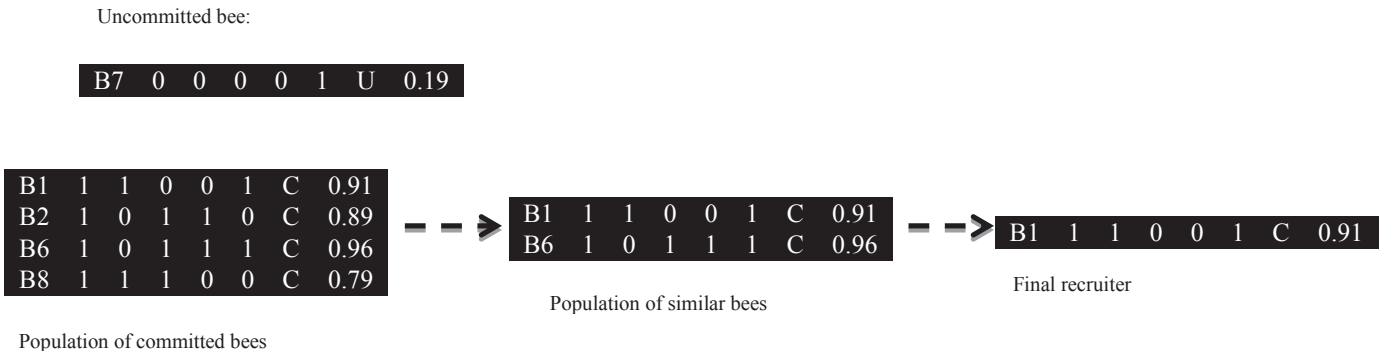

Figure 3 – Graphical illustration of stepwise filtering

4. Modeling feature selection with *w*BCO

Here the task of feature selection is modeled as a discrete (or categorical) optimization problem and the proposed *w*BCO algorithm applied, resulting in a new algorithm: FS-*w*BCO. In order to model FS as a discrete optimization task, the solutions are represented in a binary format where 1 and 0 indicate that the corresponding feature is selected or ignored, respectively. The length of the solution corresponds to the number of features of the dataset. For example a solution can be in the form of 1100010110 where the first, second, sixth, eighth and ninth features are selected only.

After initializing FS-*w*BCO, each bee takes an independent forward movement with the size of *NC*. In simpler terms, each bee decides about the selection or ignorance of the first *NC* features. As shown in Algorithm 4, at the beginning the number of remaining features (*r*) is equal to the total number of features, F, and the starting position of all the bees is set to the first feature (Lines 1 and 2). Then for *NC* number of constructive steps (the inner while loop, Lines 4 to 13) the bees create their partial solutions (inner for loop, Lines 5 to 10).

While creating the partial solutions (Lines 5 to 9), each bee takes *NC* number of constructive steps, and through taking each constructive step a bee decides whether to select a feature or not (Lines 7 to 9). *NC* is reduced by one (Line 11), as all the bees have taken their first constructive steps. After taking *NC* number of steps, the number of remaining features is reduced by *NC* (Line 14). In some cases it is likely that the number of remaining features to be less than the pre-specified *NC* (the outermost if-statement in shown in Lines 15 and 16). Hence in this case the bees will take only *r* number of steps as their *NC*. In fact *NC* is set to the number of remaining features *r* (Line 16).

Algorithm 4: Solution creation

Input:

Number of constructive steps *NC*.
Number of bees *B*.
Number of features *F*.

Output:

Created solutions.

Algorithm:

```
1.  Set the number of remaining features, r, to the total number of features F;
2.  Set the starting position of all the bees to the first feature;
3.  while the number of remaining features r is not zero
4.    while the constructive steps NC are not finished
5.      foreach bee in the population
6.        Generate a random number rand;
7.        if rand> 0.5
8.          Select the i-th feature;
          else
9.          Ignore the i-th feature;
10.     end foreach
11.     NC = NC – 1;
12.     Go to the next feature;
13.   end while
14.     r = r – NC;
15.     if r<NC
16.       r = NC;
17. end while
```

As the partial solutions are created, the bees return to their hive (the backward step) to perform three tasks. First they measure the quality of their solutions. Then bees assess the degree to which they are loyal to their partial solutions. The purpose of loyalty assessment is to divide the population into two disjoint groups of committed and uncommitted bees. Based on the *w*BCO algorithm, we use local and global weighting procedures, in which each food source is replaced with a feature in measuring the global weight of a selected feature. Finally in the third stage, each uncommitted bee will follow a committed one, according to the stepwise filtering approach.

The class labels of each sample in a dataset are considered as category labels in Equation (5). The correlation between a selected feature and the predicted class label is measured through the concept of mutual information, as in Equation (9):

$$Cor(C_p^x, F_i^j) = -\sum_p P(C_p^x) \sum_i P(f_i|C_p^x) log_2 (P(f_i|C_p^x)) \quad (9)$$

where f_i is the ith value of a feature and C_p^x is the xth predicted class label. $P(f_i|C_p^x)$ calculates the probability of co-occurrence of f_i and C_p^x. The mutuality degree between the predicted class label and the selected subset of features normalized by the summation of the mutuality degrees is the local weight of a feature. In Algorithms 5 and 6 the global and local weighting procedures of a given feature are explained. The global weight is viewed more as a ratio of the global acceptance of the feature to the total number of selected features in the population, while the local viewpoint evaluates the degree of dependency of each selected feature to the predicted class label.

In fact for each feature we preserve two sorts of information. One is global and all the bees use it in the loyalty assessment and the other is local which is a personalized weight and differs from solution to solution. Interaction of these two sorts of information in Equation (6) will identify the loyalty degree of a bee. The last step before the bees leave their hive for the next

forward movement is distinguishing follower and recruiter bees, as explained in Section 3.3. Once the recruiter bees are identified, followers will follow one and only one recruiter according to Equations (7) and (8), where all the food sources are replaced with features. Table 2 defines the parameters of the *w*BCO and FS-*w*BCO algorithm.

Algorithm 5: Global weight assessment in feature selection

Input:
A set of solutions
Output:
Global weight of the selected features f_i
Algorithm:

1. **foreach** partial solution in the population
2. **foreach** selected feature *i* of the current partial solution
3. **if** the global weight for the *i*th feature is calculated before
4. Go to the next selected feature;
else
5. Count the number of bees that have selected the *i*th feature, $f_{i.}$;
6. Sum fitness values of all the bees which have selected feature $f_{i.}$;
7. Sum fitness value of all the bees which have ignored feature $f_{i.}$;
8. Measure the global weight of the *i*th feature f_i according to Equation (2);
9. **end if**
10. **end foreach**
11. **end foreach**

Algorithm 6: Local weight assessment

Input:
A set of solutions
Output:
The local weight of each selected feature in each solution
Algorithm:

1. **foreach** solution *sol*
2. **foreach** predicted class label C_p^x
3. **foreach** selected food source F_i^j in *sol*
4. Sum the correlations between selected feature F_i^j and the predicted class label C_p^x;
5. **end foreach**
6. **foreach** selected feature F_i^j
7. Calculate the local weight of selected feature F_i^j according to Equation (5);
8. **end foreach**

Table 2 – Definition of *w*BCO and FS-*w*BCO parameters

Variable	Symbol	Initial value
Number of bees	B	User-specified
Number of generations	G	User-specified
Number of constructive steps	NC	User-specified
Number of bees selecting a food source	Sel	Problem-dependent
Number of bees ignoring a food source	$Unsel$	Problem-dependent
Fitness of a bee selected the *i*th food source	F_{Sel_i}	Problem-dependent
Fitness of a bee ignored the *i*th food source	F_{Unsel_i}	Problem-dependent
The *x*th predicted class label	C_p^x	Problem-dependent
*j*th food source of the *i*th bee	F_i^j	Problem-dependent
Number of in-common food sources	E_i	Problem-dependent
Fitness of the *cb*th committed bee	F_{com}^{cb}	Problem-dependent
Fitness of an uncommitted bee	F_{uncom}	Problem-dependent
*i*th value of F_i^j	f_i	Problem-dependent
Number of remaining features	r	Problem-dependent

5. Experimental results

In this section we empirically investigate the effectiveness of *w*BCO and FS-*w*BCO. Initially we describe the datasets and functions used in these experiments. Then some numerical experiments will be carried out to investigate the efficacy of the proposed algorithms. The parameter setting of the proposed algorithms and the competitors are also explained in a separate section. As we primarily claim proposing improvements over BCO, we conduct convergence behavior experiments of *w*BCO and compare the results against conventional BCO. The Wilcoxon statistical test is used to investigate statistically the performance of the proposed variations. In summary this section aims at finding the answers to the following questions:

1) Are the proposed modifications good enough to enhance the convergence behavior of conventional BCO?
2) Does the number of bees have any effect on the loyalty assessment and stepwise filtering assignment stages of *w*BCO?
3) *w*BCO targets the exploitation power of BCO, but under what circumstances is it good enough to compete with other BCO variations that enhance both exploration and exploitation powers?
4) What is the main influential factor which can affect the performance of *w*BCO in feature selection?
5) How does FS-*w*BCO compare to other swarm and evolutionary FS algorithms? Can the algorithm outperform all competitors for all datasets?

5.1. Datasets and benchmark functions

In this section we introduce the benchmark datasets and functions used for the experiments of *w*BCO and FS-*w*BCO. For *w*BCO, the performance is tested using the standard benchmark functions as shown in Table 3. These functions are accessible from different resources including Simon Fraser University benchmark function repository[1], and also the work of Kang and colleagues [**12**]. In the experiments of *w*BCO, since the proposed algorithms are applicable only for discrete optimization problems, the functions' inputs are in integer format.

[1] http://www.sfu.ca/~ssurjano/optimization.html

Table 3 – The benchmark functions

Function Name		Equations	Intervals	*N*	Global minimal	Class
f_1	Levy (N.13)	$sin^2(3\pi x) + (x-1)^2\left(1+sin^2(3\pi y)\right) + (y-1)^2(1+sin^2(2\pi y))$	-10≤ x, y ≤10	2	F(x*)=0	Small
f_2	Bohachevsky 1	$x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	-10≤ x_1, x_2≤10	2	F(x*)=0	
f_3	Booth's	$(x+2y-7)^2 + (2x+y-5)^2$	-10≤ x, y ≤ 10	2	F(x*)=0	
f_4	Hartman 3	$\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	0≤ x_i ≤ 1	3	F(x*)= -3.862782	
f_5	Matyas	$0.26(x^2+y^2) - 0.48xy$	-10≤x,y≤10	2	F(x*)=0	
f_6	Scaffer's F6	$0.5 + \frac{(sin^2(\sqrt{x^2+y^2}) - 0.5)}{(1+0.001(x^2+y^2))^2}$	-5≤x,y≤5	2	F(x*)=0	
f_7	Schwefel's 1.2	$\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	-10 ≤x_i≤ 10	10	F(x*) = 0	Medium
f_8	Michalewicz	$-\sum_{i=1}^{d} \sin(x_i)\, sin^{2m}\left(\frac{ix_i^2}{\pi}\right) \qquad m = 10$	-5≤ x_i ≤ 5	10	F(x*)= -9.66015	
f_9	Trid	$\sum_{i=1}^{n}(x_i - 1)^2 - \sum_{j=2}^{i}(x_i x_{i+1})$	-15 ≤x_i≤ 15	15	F(x*) = -200	
f_{10}	Schwefel's 2.22	$\sum_{i=1}^{N} \lvert x_i \rvert + \prod_{i=1}^{N} \lvert x_i \rvert$	-10≤ x_i ≤ 10	20	F(x*)=0	
f_{11}	Bohachevsky 2	$\sum_{i=1}^{n-1} x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i)\cos(4\pi x_{i+1}) + 0.3$	-5≤ x_i ≤ 5	20	F(x*)=0	
f_{12}	Expansion F10	$f(x) = f_{10}(x_1, x_2) + \cdots + f_{10}(x_n, x_0)$ $f_{10}(x,y) = (x^2+y^2)^{0.25} \times [sin^2(50 \times (x^2+y^2)^{0.1}) + 1]$	-100≤ x_i ≤ 100	25	F(x*)=0	
f_{13}	Styblinski	$\frac{1}{2}\sum_{i=1}^{N}(x_i^4 - 16x_i^2 + 5x_i)$	-5≤ x_i ≤ 5	30	F(x*)= -39.16599N	Large
f_{14}	Penalized 1	$\frac{\pi}{n}\left\{10sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1}) + (y_n-1)^2]\right\} + \sum_{i=1}^{n-1} u(x_i, 10, 100, 4)$ $y_i = 1 + 0.25(x_i+1)$ *and* $u = \begin{cases} k(x_i-a)^m & x_i > a \\ 0 & -a \le x_i < a \\ k(-x_i-a)^m & x_i < -a \end{cases}$	-5≤ x_i ≤ 5	30	F(x*)=0	
f_{15}	Weierstrass	$\sum_{i=1}^{N}\sum_{k=0}^{20}[(0.5)^k\cos(2\pi 3^k(x_i+0.5))] - N\sum_{k=0}^{20}[(0.5)^k\cos(2\pi 3^k \times 0.5)]$	-10≤ x_i ≤ 10	35	F(x*)=0	
f_{16}	Step	$\sum_{i=1}^{N} ABS(x_i + 0.5)^2$	-100≤ x_i ≤ 100	40	F(x*)=0	
f_{17}	Rotated Rastrigin	$\sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	-20≤ x_i ≤ 20	45	F(x*)=0	
f_{18}	Penalized 2	$0.1\left\{sin^2(3\pi y_1) + \sum_{i=1}^{n-1}(x_i-1)^2[1+sin^2(3\pi x_{i+1}) + (x_n-1)^2]\right\} + \sum_{i=1}^{n-1} u(x_i, 5, 100, 4)$	-5≤ x_i ≤ 5	45	F(x*)=0	

In the experiments of FS-*w*BCO, we used several standard benchmark datasets as shown in Table 4. The datasets are from the UCI machine learning repository[2]. The preferred approach is to train the model with a set of samples and then test the trained model using samples which have never been used in training procedure. Therefore hold-out evaluation procedure is used for experiments, in which the dataset is divided into three subcategories of training, testing, and validation. We divided a given dataset into 70 percent training, 20 percent validation data, and 10 percent testing data.

Table 4 - Dataset details

Datasets	(Training /Test/Validation)	# Features	# Classes	NC
Breast Cancer (BC)	(490/42/167)	9	2	3
Glass (GL)	(150/13/61)	10	7	2
Sonar (SO)	(146/12/50)	60	2	12
Horse (HR)	(195/17/67)	27	2	9
Wisconsin Breast Cancer (WDBC)	(398/34/137)	30	2	10
Vehicle (VC)	(592/52/202)	18	4	6
MUSK 1(MUS 1)	(272/60/144)	168	2	35
Arrhythmia (ARR)	(316/27/109)	279	16	50

5.2. Parameter setting

In comparisons of *w*BCO some algorithms are implemented, including BCO [**4**] and other bee colony-based variations such as *Disr*BCO [**24**], chaotic differential BCO (CDBCO) [**21**], and DBCO [**23**]. The parameters of the proposed and competitor algorithms are set either based on the reference papers or empirical studies. Table 5 outlines the parameter settings of *w*BCO and its competitors. For the algorithms CDBCO, *Disr*BCO, and DBCO, the parameter setting is done according to the reference papers, while in BCO and *w*BCO parameters are fine-tuned to their most optimal values. In order to make justifiable comparisons, the number of iterations and the population sizes are identical for all the algorithms. *NC* in *w*BCO experiments is set empirically to D/4, 2D/4, and 3D/4 for small, medium, and large functions, respectively and in FS-*w*BCO experiments it is set to different values for different datasets as shown in Table 4. Also in order to investigate the effect of population size on the performance of *w*BCO (the second experimental question) the swarm size of the bees is set to three different values of 50, 500, and 5000.

Table 5 – Parameter setting for numerical experiments

Algorithm	Parameters
*w*BCO	N= variable; NC= variable; NI= 100.
BCO	
*Disr*BCO	N= variable; NC= variable; NI= 100; committed bees = 0.9N; Uncommitted bees = 0.1N.
CDBCO	N= variable; NI = $Cycle_{max} = 100$; $K_{max} = 20$; $Limit_{abandon} = 8$; C = 0.3; F = 0.3.
DBCO	N= variable; NI= 100; $\alpha = 1$; $\gamma = 1$; $\beta = 1/2$; $\sigma = 1/2$.

In the experiments involving FS-*w*BCO, we implemented the competitor algorithms and their settings are determined based on the reference paper as shown in Table 6. Here, we are comparing against different algorithms including BCOFS [**7**], *Dis*ABC [**31**], BPSO [**8**] and the evolutionary algorithms RHS [**32**] and HGAFS [**33**]. *Dis*ABC was originally proposed as an

[2] Datasets can be downloaded from:
https://drive.google.com/folderview?id=0B5R8ibfLZ7zmS3Q1bTFHb3NZdWM&usp=sharing

optimization algorithm for binary functions. In this paper we implemented it as a feature selector algorithm. We use SVM for classification, and for this purpose the implementation of Mathew Johnson[3] version 1.6.3 is used. This implementation is publically available in C# and we set the dynamic parameters according to the implementation.

In BCO and FS-*w*BCO algorithms, the number of constructive steps (*NC*) is set to values which ensure reaching optimal solutions and adjusted empirically for each dataset according to Table 6. In BPSO, *k* is the desired subset size. In the setting of *k*, for datasets that are in common with our work, we used the same values as reported in the reference paper, while for the other datasets the desired subset size is set to the same subset size gained by FS-*w*BCO.

Table 6 - Parameter setting of the implemented FS algorithms

Category	Algorithm	Parameter settings
Proposed	FS-*w*BCO	B = 5000, NC = depends on the dataset size.
Swarm based	BCOFS	
	BPSO	$Population\ size = 5000, \omega_{min} = 0.5, \omega_{max} = 0.995$ $c_1 = 2, c_1 = 1.5, c_3 = 0.5, k = variable.$
	*Dis*ABC	$population\ size = 5000, \varphi_{min} = 0.5, \varphi_{max} = 0.9$ $P_s = 1, P_{local} = 0.02, N_{local} = 100$
Evolutionary based	RHS	HMS = 5000, HMCR = 0.65, PAR_{Min} = 0.5, PAR_{Max} = 0.9.
	HGAFS	Pool_size = 5000, mutation_rate = 0.02, crossover_rate = 0.6.

The algorithms (i.e. *w*BCO and FS-*w*BCO) are executed for several iterations. The executions lower than 30 showed significant changes to the results of algorithms while in executions with 30 or higher slight changes were seen in the final results. Therefore we set the number of independent executions to 30. In each execution we set the number of iterations to 100 as in iterations higher than 100 only slight changes were seen in the final results.

5.3. Convergence behavior studies

In this section, we investigate the convergence behavior of BCO and *w*BCO, to address the first question. The convergence behavior experiments in this section will show how many bees are required at least for an algorithm to reach its near optimal solution. We refer to near optimal solutions as according to [**29**] there is no specific algorithm to regularly achieve the best solution for all optimization problems. In population-based algorithms such as *w*BCO and BCO where there is no connection between each pair of consecutive iterations, the number of bees will have an impact on the convergence rather than number of iterations. Hence increasing or decreasing the population size according to the solution space size will show how effective *w*BCO and BCO are in reaching near optimal solutions.

In Figure 4, the Matyas function is selected as an example of a small size function. This function is used as the experiments indicated that it is very sensitive to even small changes of the algorithms' setting. Therefore it can better reflect the convergence speed of the algorithms. *w*BCO converges when the number of bees is 300 or higher, while BCO requires more bees to obtain convergence. More bees will impose a higher execution time. Hence, *w*BCO can gain an optimal result quicker compared to BCO. In Figures 5 and 6, when increasing the size of the

[3] http://www.matthewajohnson.org/software/svm.html

solution space, the number of bees needed for exploration increases. In Michalewicz and Penalized 1 as examples of medium and large functions, respectively, the same scenario as in small functions occurs, in the sense that *w*BCO has better convergence in comparison to BCO, thanks to the applied modifications.

When increasing the size of the solution space and correspondingly the number of bees, *w*BCO shows less chaotic behavior. In Figure 4, *w*BCO is quite variable before convergence, while in Figures 6 and 7 this behavior reduces quickly before convergence occurs. This fact indicates that in *w*BCO the larger the number of bees, the more information is collected regarding the food sources in the surrounding area of the hive. Hence, the algorithm can move the bees around the near optimal parts from the early stages of algorithm execution to finally enhance the convergence behaviour of conventional BCO.

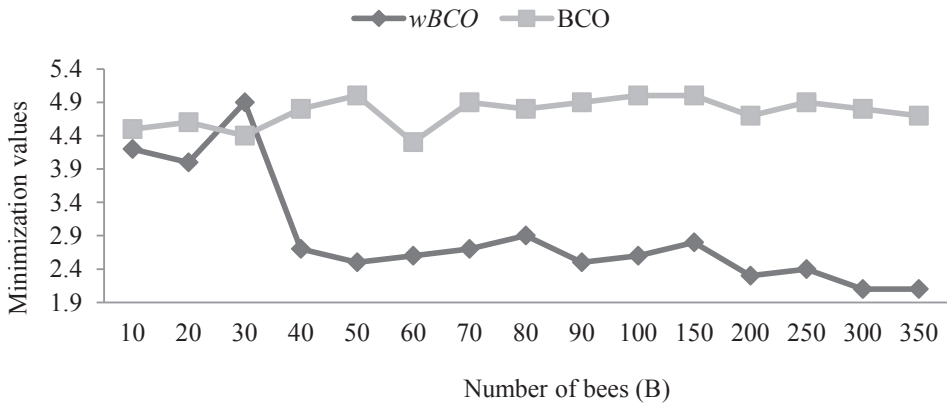

Figure 4 - Convergence behavior studies on Matyas as a sample of small functions in population-based scenario

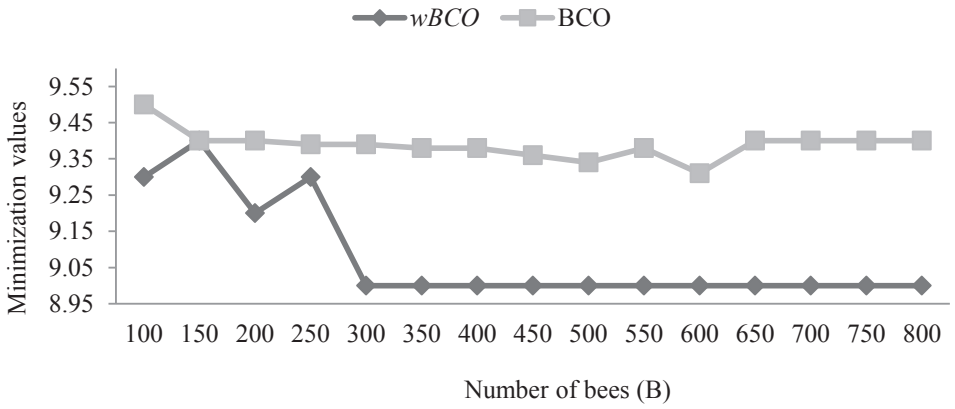

Figure 5 - Convergence behavior studies on Michalewicz as a sample of medium functions in population-based scenario

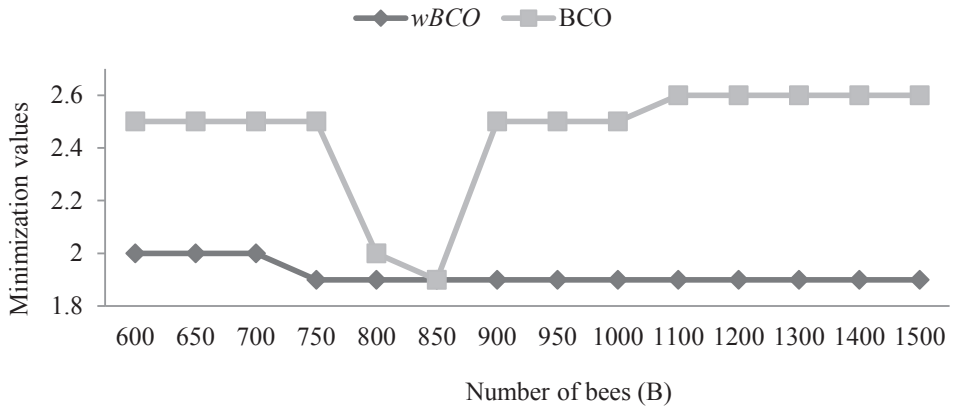

Figure 6 - Convergence behavior studies on $_{Penalized\ 1}$ as a sample of large functions in population-based scenario

5.4. Performances and comparisons

In this section, the performance of FS-*w*BCO and *w*BCO are measured. In the experiments of *w*BCO we are using 18 minimization functions as shown in Table 3 and relevant comparisons are made with other bee colony-based approaches. As explained before, ABC and BCO are two different algorithms. But as ABC relies on the natural behavior of the bees in locating food sources we consider ABC as another relevant work for comparison. One may ask: once the bees' loyalty degrees are assessed, how will they be divided into two disjoint groups of committed and uncommitted bees? The bees can be divided based on different division strategies. In this paper, and according to our experimentation, we divide the bees based on the average values of their loyalty degrees, where bees having loyalty degrees higher than the average are considered to be loyal.

It is important to note that the proposed algorithms are suitable only for discrete optimization; continuous optimization algorithms are not relevant here and cannot be compared. This restricts the algorithms to choose only integer values. Hence, the competitors are implemented and tested in our setting and experimental conditions.

Table 7 shows the experiments of *w*BCO and its competitors with three different values for population size. The results indicate that the performance of *w*BCO is sensitive to the solution space size and the number of bees. The higher the number of bees, the better *w*BCO performs. This is as a result of more information which is available in order to measure the loyalty degrees and assess the stepwise filtering assignment.

When the solution space is small (i.e. in functions F_1 to F_6), setting $B = 50$ produces a performance that is better than the competitors. By increasing the size of the solution space this superiority declines. Once the size of the solution space gets larger more possible solutions will be available and consequently more bees will be required to explore the solution space adequately. The convergence behavior experiments also demonstrate this fact. Hence by increasing the population size, *w*BCO performs better as it will be able to measure the loyalty degrees in a more effective way, thanks to the availability of more information. This information will also help the bees to choose better followers in the stepwise filtering assignment step. Therefore the answer to the second experimental question is positive.

The *w*BCO algorithm targets the enhancement of the exploration power of BCO, and it is expected to be not as effective as algorithms that enhance both the exploration and exploitation of BCO, such as DBCO and CDBCO. The experiments in Table 7 indicate that through increasing the number of bees, the amount of available information can be handled efficiently by loyalty assessment operations and stepwise filtering assignment, to outperform CDBCO and DBCO. However in larger populations there is still some inferiority in the performance of *w*BCO. The proposed algorithm is inferior to DBCO and CDBCO for the two functions F_9 and F_5.

As an answer to the third question, through increasing the population size in *w*BCO it is likely that the algorithms that enhance both the exploration and exploitation power of BCO will be outperformed. *Disr*BCO, similar to *w*BCO, tries to enhance the exploitation power of BCO, but this algorithm is more effective than *w*BCO only for the F_{12} function.

Table 7 – Evaluations using minimization functions with different population sizes

			B=50					B=500					B=5000				
			*W*BCO	BCO	*Disr*BCO	CDBCO	DBCO	*W*BCO	BCO	*Disr*BCO	CDBCO	DBCO	*W*BCO	BCO	*Disr*BCO	CDBCO	DBCO
Small	F_1	SD	0.33E+2	0.43E+2	0.47E+2	0.87E+2	0.43E+2	0.43E+2	0.49E+2	0.46E+2	0.76E+2	0.25E+2	0.44E+2	0.47E+2	0.49E+2	0.83E+2	0.486E+2
		Avg	**0.44E+2**	0.68E+2	0.61E+2	0.61E+2	0.67E+2	0.47E+2	0.7E+2	0.7E+2	0.59E+2	**0.43E+2**	**0.43E+2**	0.7E+2	0.71E+2	0.66E+2	0.717E+2
	F_2	SD	0.35E+2	0.6E+2	0.59E+2	0.68E+2	0.66E+2	0.5E+2	0.68E+2	0.63E+2	0.63E+2	0.23E+2	0.55E+2	0.66E+2	0.67E+2	0.8E+2	0.66E+2
		Avg	**0.44E+2**	1.01E+2	0.79E+2	0.74E+2	0.88E+2	0.53E+2	0.96E+2	0.9E+2	0.61E+2	**0.41E+2**	**0.59E+2**	0.98E+2	1.08E+1	0.665E+2	E+1
	F_3	SD	0.23E+3	0.15E+3	0.31E+3	0.4E+3	0.34E+3	0.26E+3	0.3E+3	0.38E+3	0.48E+2	0.48E+3	0.23E+3	0.22E+2	0.5E+3	0.46E+3	0.507E+3
		Avg	**0.2E+3**	0.21E+3	0.28E+3	0.24E+3	0.31E+3	**0.21E+3**	0.3E+3	0.38E+3	0.26E+4	0.43E+2	**0.19E+3**	0.27E+3	0.44E+3	0.352E+3	0.452E+3
	F_4	SD	0.41E+0	0.64E+0	0.77E+0	0.68E+0	1.04E+0	0.48E+0	0.57E+0	0.73E+0	0.63E+0	0.93E+0	0.57E+0	0.6E+0	0.92E+0	0.9E+0	0.95E+0
		Avg	**0.3E+0**	0.44E+0	0.72E+0	0.63E+0	1.1E+0	**0.38E+0**	0.534E+0	0.82E+0	0.52E+0	0.88E+0	**0.38E+0**	0.6E+0	0.85E+0	0.54E+0	0.86E+0
	F_5	SD	0.99E+1	0.13E+2	0.14E+2	0.7E+0	0.2E+2	0.11E+3	1.7E+1	0.16E+3	1.05E+0	1.91E+1	0.82E+1	0.19E+2	0.18E+2	0.99E+0	0.188E+2
		Avg	0.88E+1	1.2E+1	1.42E+1	**0.11E+1**	0.15E+2	0.82E+1	1.5E+1	0.14E+2	**0.9E+0**	1.7E+1	0.6E+1	1. 7E+1	1. 6E+1	**0.7E+0**	1. 73E+1
	F_6	SD	0.27E+0	0.23E+0	0.3E+0	0.3E+0	0.33E+0	0.31E+0	0.18E+0	0.34E+0	0.27E+0	0.33E+0	0.32E+0	0.19E+0	0.33E+0	0.31E+0	0.335E+0
		Avg	**0.22E+0**	0.26E+0	0.57E+0	0.71E+0	0.51E+0	**0.24E+0**	0.25E+0	0.53E+0	0.65E+0	0.52E+0	**0.27E+0**	0.27E+0	0.53E+0	0.62E+0	0.533E+0
Medium	F_7	SD	0.12E+2	0.1E+2	0.8E+4	0.91E+5	0.29E+5	0.2E+4	0.6E+4	0.1E+4	7.84E+4	3.34E+4	1.84E+3	16.5E+3	13.2E+3	97E+3	34.8E+3
		Avg	**0.81E+2**	0.97E+2	0.51E+4	0.55E+5	0.24E+5	**0.14E+4**	0.24E+5	0.61E+4	6.45E+4	2.72E+4	**1.35E+3**	5.65E+3	12.9E+3	60.3E+3	27.4E+3
	F_8	SD	0.25E+0	0.42E+0	0.48E+1	0.76E+0	0.55E+0	0.81E+0	0.34E+0	0.47E+1	0.58E+0	0.58E+0	1.13E+0	0.4E+0	0.38E+0	0.41E+0	0.258E+0
		Avg	0.89E+1	0.94E+1	**0.51E+1**	0.1E+2	0.96E+1	0.9E+1	0.94E+1	**0.47E+1**	0.98E+1	0.97E+1	**0.86E+1**	0.9E+1	0.95E+1	0.101E+2	0.881E+1
	F_9	SD	0.11E+3	0.32E+3	0.18E+3	0.38E+3	0.38E+3	0.2E+3	0.32E+3	0.23E+3	0.93E+2	0.36E+3	0.36E+3	0.3E+3	0.3E+3	0.93E+2	0.385E+3
		Avg	0.49E+3	0.96E+3	**0.12E+3**	0.47E+3	0.13E+4	0.46E+3	0.93E+3	1.15E+3	**0.28E+3**	1.31E+3	0.54E+3	0.95E+3	1.22E+3	**0.28E+3**	1.32E+3
	F_{10}	SD	0.12E+2	0.2E++3	0.1E+2	0.39E+2	0.11E+2	0.23E+2	0.11E+2	0.12E+2	0.39E+2	0.12E+2	0.28E+2	0.11E+3	0.12E+2	0.44E+2	0.132E+2
		Avg	**0.26E+2**	0.63E+3	0.95E+2	0.11E+3	0.1E+3	**0.62E+2**	0.96E+2	0.97E+2	1.06E+2	0.99E+2	**0.56E+2**	0.98E+2	0.99E+2	1.02E+2	E+2
	F_{11}	SD	0.89E+2	0.11E+3	0.23E+3	0.17E+3	0.1E+3	0.14E+3	0.1E+3	0.08E+3	0.32E+3	0.1E+3	0.15E+3	0.99E+2	0.9E+2	0.21E+3	0.1E+3
		Avg	0.3E+3	0.48E+3	**0.23E+3**	0.62E+3	0.48E+3	**0.37E+3**	0.47E+3	0.44E+3	0.5E+3	0.48E+3	**0.34E+3**	0.48E+3	0.46E+3	0.6E+3	0.47E+3
	F_{12}	SD	0.18E+2	0.56E+3	0.15E+3	0.94E+3	0.47E+2	0.19E+2	0.54E+2	1.48E+2	0.94E+2	0.5E+2	0.2E+2	0.52E+2	0.15E+3	0.89E+2	0.5E+2
		Avg	0.21E+3	0.32E+3	**0.17E+3**	0.28E+3	0.31E+3	0.21E+3	0.31E+3	**0.14E+3**	0.29E+3	0.32E+3	0.2E+3	0.31E+3	**0.15E+3**	0.3E+3	0.32E+3
Large	F_{13}	SD	0.17E+3	0.21E+3	0.51E+3	0.38E+3	0.26E+3	0.26E+3	0.18E+3	0.14E+3	0.234E+3	0.2E+3	0.31E+3	0.2E+3	0.17E+3	0.3E+3	0.114E+3
		Avg	0.75E+3	0.1E+4	**0.66E+3**	1.7E+3	0.1E+4	**0.8E+3**	1.07E+3	1.02E+3	1.98E+3	1.1E+3	**0.78E+3**	1.1E+3	1.06E+3	1.916E+3	0.962E+3
	F_{14}	SD	0.39E+0	0.38E+0	0.13E+1	0.26E+1	0.41E+0	0.56E+0	0.38E+0	0.38E+0	0.22E+1	0.6E+0	0.79E+3	0.34E+1	0.26E+1	0.285E+3	0.527E+0
		Avg	0.2E+1	0.27E+1	**0.14E+1**	0.32E+1	0.3E+1	**0.2E+1**	0.26E+1	0.26E+1	0.38E+1	0.28E+1	**0.19E+1**	0.25E+1	0.28E+1	0.32E+1	2.86E+1
	F_{15}	SD	9.8E-12	1.3E-11	2.9E-11	0.24E+2	1.6E-11	1.19E-11	8.7E-12	1.09E-11	0.29E+2	1.4E-11	1.7E-11	E-11	1.08E-11	0.278E+2	1.4E-11
		Avg	4.7E-11	4.06E-11	**3.7E-11**	0.17E+3	6.5E-11	**5.54E-11**	5.57E-11	6.06E-11	0.176E+3	6.6E-11	**5.52E-11**	5.74E-11	6.05E-11	0.173E+3	6.5E-11
	F_{16}	SD	0.1E+4	0.84E+3	6.3E+4	0.11E+6	0.19E+5	0.13E+4	0.84E+3	0.63E+4	110.8E+3	19.3E+3	1.82E+3	1.01E+3	14.3E+3	120E+3	18.6E+3
		Avg	**0.61E+4**	0.73E+4	6.3E+4	0.14E+6	0.13E+6	**5.8E+3**	7.3E+3	6.3E+3	142.4E+3	133.3E+3	**5.57E+3**	7.13E+3	126E+3	134E+3	133.2E+3
	F_{17}	SD	0.52E+3	0.89E+3	2.9E+3	0.55E+4	0.78E+3	0.1E+4	0.66E+3	0.62E+4	5.17E+3	0.57E+3	1.26E+3	0.7E+3	0.64E+3	5.33E+3	0.8E+3
		Avg	0.47E+4	0.49E+4	**0.3E+4**	0.65E+4	0.59E+4	**0.47E+4**	0.57E+4	0.56E+4	0.64E+4	0.56E+4	**4.5E+3**	5.2E+3	5.78E+3	6.4E+3	6E+3
	F_{18}	SD	0.1E+2	0.12E+2	0.22E+2	0.46E+2	0.84E+1	0.12E+2	0.07E+2	0.58E+1	0.51E+2	0.72E+1	0.146E+2	0. 63E+1	0.07E+2	0.32E+2	0.75E+2
		Avg	0.37E+2	0.41E+2	**0.26E+2**	0.54E+2	0.49E+2	**0.34E+2**	0.46E+2	0.44E+2	0.61E+2	0.47E+2	**0.31E+2**	0.46E+2	0.47E+2	0.75E+2	0.472E+2

Another set of experiments that reveal the inferiorities and superiorities of FS-*w*BCO have been conducted and the results are shown in Table 8. The population size is set to 5000 as *w*BCO showed the best performance in this setting. For each iteration, the best value is selected, and for each execution the 100 best values are averaged, and finally the results are averaged over 30 executions. In this table two well-known measures, classification accuracy (CA) and kappa statistic (KS), are used to evaluate the performance of FS-*w*BCO. CA is given in Equation (10), where *#TS* and *#TC* are the total number of samples and correctly classified samples, respectively.

$$CA = \frac{\#TC}{\#TS} \tag{10}$$

The other measure is the kappa statistic [**34**], which is a prevalent statistical measure that allows for input sampling bias. This measure has been used in many algorithms [**35**] [**36**] [**37**] [**6**] and is calculated via Equations (11) to (13). The aim of using this measure is to assess the level of agreement between the classifier's output and the actual classes of the dataset. The kappa statistic is calculated as follows:

$$P(A) = \frac{\sum_{i=1}^{c} N_i}{N} \tag{11}$$

$$P(E) = \sum_{i=1}^{c} \left(\frac{N_{i*}}{N} \cdot \frac{N_{*i}}{N}\right) \tag{12}$$

$$KS = \frac{P(A)-P(E)}{1-P(E)} \tag{13}$$

where c is the category/class label, N is the total number of samples and N_i is the number of samples in a dataset which are correctly classified by the classifier. In Equation (12) N_{i*} is the number of instances recognized as class i by the classifier and N_{*i} is the number of instances that belong to class i in the dataset. The purpose is to maximize this measure. Finally, kappa (or in short *KS*) is measured in Equation (13) in which $kappa \in [0, 1]$, where *kappa* = 0 and *kappa* = 1 means there is no agreement between the classifier and the actual classes, and perfect agreement on every example, respectively.

In Table 8, FS-*w*BCO is compared against other evolutionary and swarm-based feature selection algorithms to investigate its performance. The experiments are divided into two parts of testing and training. The purpose of comparisons and experiments with the training data is to investigate how effective the algorithms are in model creation. Once the model is constructed, the testing data is used to evaluate the effectiveness of the proposed algorithm. In the training experiments, we execute the algorithms with training and validation sets and then preserve the best solutions for each iteration. In the testing experiments, we retrieve the preserved solutions and apply them to the testing set of each dataset. The results are averaged over all iterations. KS values of training results are shown in the form of x(y), in which x and y are the average values of kappa and the best subsets, respectively.

In the experiments, FS-*w*BCO is compared with swarm-based algorithms including BCOFS [**7**], *Dis*ABC [**31**], and BPSO [**8**]. As the results for the training data indicate, in comparing

BCOFS and FS-*w*BCO, the proposed algorithm performs better than BCOFS in most of the datasets. The other set of comparisons is made with the evolutionary algorithms RHS [**32**] and HGAFS [**33**]. Also in the training results, for the CA measure, the improvement in performance of FS-*w*BCO is less significant compared to RHS [**32**]. Additionally, FS-*w*BCO performs better than or similar to HGAFS. For the KS measure, FS-*w*BCO is inferior to RHS and mostly better than HGAFS. The inferiorities could be as a result of the hold-out strategy as different types of division might lead to slightly different results. Even though this will address the fourth question, however, the preferred approach is to test the algorithm with samples that have never been used in model creation, as used in other feature selection works [**8**] [**33**].

To address the fifth question, in the testing dataset comparisons, *w*BCO is superior to conventional BCO in general. Gaining this superiority is the primary purpose of *w*BCO, thanks to a reliance on stepwise assignment filtering and loyalty measurement policies. In comparisons with other swarm-based algorithms, FS-*w*BCO mainly has similar performance compared to BPSO, with superiorities in a few datasets. This could be as a result of the nature of the algorithms in the sense that one is PSO and the other is BCO-based. In comparisons to *Dis*ABC and evolutionary-based variations with FS-*w*BCO, the proposed algorithm could outperform competitors.

Table 8 – KS and CA results for training and testing data using the SVM classifier [values in the range 0-1]

		Classification Accuracy						Kappa Statistics					
		Proposed	Swarm based			Evolutionary based		Proposed	Swarm based			Evolutionary based	
		FS-*w*BCO	BCOFS	*Dis*ABC	BPSO	RHS	HGAFS	FS-*w*BCO	BCOFS	*Dis*ABC	BPSO	RHS	HGAFS
Training results	BC	0.95	**0.97**	0.95	0.96	0.96	0.938	0.917(6)	**0.96(3)**	0. 95(7)	**0.96(7)**	**0.96(2)**	0.938(2)
	GL	**0.935**	**0.935**	0.927	**0.935**	0.921	**0.935**	0.812 (7)	0.843 (6)	0.781 (5)	0.812 (7)	**0.856 (4)**	0.812 (2)
	SO	0. 785	0.635	0.69	0.762	**0.88**	0.738	0.78(27)	0.66(22)	0.682(33)	0.756 (27)	**0.878 (28)**	0.731 (4)
	HR	**0.871**	0.731	0.79	0.82	**0.871**	0.846	**0.868(12)**	0.642(29)	0.795 (14)	0.81 (12)	**0.868(2)**	0.842(2)
	WDBC	0.946	0.937	0.933	**0.95**	0.945	**0.95**	0.945(16)	0.93 (15)	0.933 (15)	0.94(7)	0.945 (7)	**0.95 (3)**
	VC	0.9	0.864	0.88	0.914	**0.91**	0.868	0.8 (11)	0.668 (11)	0.76 (10)	0.83 (11)	**0.836 (14)**	0.736 (6)
	MUS 1	0.826	0.77	0.798	0.826	**0.833**	0.798	0.82 (76)	0.82 (72)	0.797 (57)	0.825 (76)	**0.832 (57)**	0.797 (35)
	ARR	**0.67**	0.655	0.65	0.65	0.653	0.655	**0.66 (148)**	0.588 (135)	0.624 (87)	0.63 (148)	0.63 (133)	0.63 (19)
Testing results	BC	**0.981**	**0.981**	0.976	**0.981**	0.971	0.976	**0.98**	**0.98**	0.976	**0.98**	0.971	0.976
	GL	**0.962**	0.952	0.952	0.934	0.934	0.915	**0.961**	0.952	0.952	0.933	0.933	0.914
	SO	0.644	0.644	0.66	**0.677**	0.593	0.593	0.637	0.637	0.65	**0.672**	0.586	0.586
	HR	0.835	0.717	0.823	0.776	0.823	**0.847**	0.833	0.678	0.821	0.773	0.821	**0.845**
	WDBC	**1**	0.965	0.965	0.965	0.965	0.965	**1**	0.964	0.964	0.964	0.964	0.964
	VC	**0.93**	0.918	**0.93**	**0.93**	0.918	0.883	**0.863**	0.84	0.863	0.863	0.84	0.772
	MUS 1	**0.916**	0.833	0.883	**0.916**	0.9	0.9	**0.915**	0.83	0.881	**0.915**	0.898	0.898
	ARR	**0.67**	0.658	0.658	0.658	0.658	0.656	**0.66**	0.652	0.652	0.652	0.652	0.637

5.5. Wilcoxon statistical test

In order to show that both *w*BCO and FS-*w*BCO are comparable to the state-of-the-art algorithms for the unseen data, we measure their performance statistically using the Wilcoxon signed-ranks test. According to [**38**] this measure is more sensible than the t-test as it assumes commensurability of differences, but only qualitatively. In other words, greater differences still count more, which is probably desired, but the absolute magnitudes are not considered. The test is safer since it does not assume normal distributions, and outliers have less effect. The Wilcoxon signed-ranks test [**39**] is a non-parametric statistical measure which ranks the differences in performances of two algorithms on each data set. The test is calculated using Equation (14):

$$V = \min(S^+, S^-) \tag{14}$$

where values of S^- and $S^+$are measured using Equations (15) and (16), repectively.

$$S^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{15}$$

$$S^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{16}$$

Here, $d_i$is the difference between the performance scores of two algorithms on *i*th out of *D* datasets. Ranks of $d_i = 0$ are split equally among the sums, if there is an odd number of $d_i = 0$ then one is ignored. Then, using the critical values and the values gained for Wilcoxon test (V) it can be inferred if the null hypothesis is rejected or not.

According to Table 9 and the table of exact critical values, it can be inferred if the null hypothesis is rejected or not. Once a null hypothesis is rejected, it means that the difference between two conditions is unlikely to have occurred by chance. In the experiments of *w*BCO, 18 functions are used (i.e. N =18) and we are considering the results of *w*BCO with $B = 5000$, as it showed the best performance.

For a confidence level of $\alpha = 0.01$, V values less than 40 can reject the null hypothesis. Therefore *w*BCO has already rejected the null hypothesis in relation to its competitors. Similarly in other confidence levels of $\alpha = 0.02$ and $\alpha = 0.05$, where the null hypothesis can be rejected for different values of 33 and 40, respective, the null hypothesis is rejected.

Table 9 – Wilcoxon test of *w*BCO and the competitors

		Second Algorithm				
		*w*BCO	BCO	*Disr*BCO	CDBCO	DBCO
First Algorithm	*w*BCO		0.5	1	6	0
	BCO	0.5		19	34	38
	*Disr*BCO	1	19		52	19
	CDBCO	6	34	52		48
	DBCO	0	38	19	48	

Table 10 shows the results of Wilcoxon values (V) of FS-*w*BCO. In these experiments we are using eight datasets (i.e. $N = 8$). Once the confidence level is 0.01, the null hypothesis could not be rejected, unless in relation to RHS algorithm. In other confidence levels of $\alpha = 0.02$ and $\alpha = 0.05$ the null hypothesis is rejected for BCO, HGAFS and RHS algorithms.

Table 10 - Wilcoxon test of FS-*w*BCO and the competitors

		Second Algorithm					
		FS-*w*BCO	BCO	FS-*Dis*ABC	BPSO	RHS	HGAFS
First Algorithm	FS-*w*BCO		1	5	4.5	0	2
	BCO	1		3.5	4	10	17
	FS-*Dis*ABC	5	3.5		8.5	5.5	8
	BPSO	4.5	4	8.5		5.5	6
	RHS	0	10	5.5	5.5		7.5
	HGAFS	2	17	8	6	7.5	

6. Conclusion and future work

BCO is a swarm-based optimization algorithm that is good at exploration but somewhat weak concerning exploitation. In order to improve the exploitation power of BCO, we proposed a novel algorithm called *w*BCO which considers the weights of traversed food sources. For each food source, two weights are considered: one is global that identifies the overall popularity of a food source in the swarm and the other is local which indicates the extent to which the selected food source is correlated to the category labels.

In line with the improvements of exploitation power, we adopted a new recruiter selection procedure which assigns an uncommitted bee to the most similar committed ones. In order to investigate the utility of *w*BCO we applied it to the FS area and introduce FS-*w*BCO. The efficiency of *w*BCO was measured through some of the well-known benchmark functions and relevant comparisons were made with other bee colony-based algorithms.

The results show that *w*BCO is sensitive to the solution space size; by growing the size of the solution space, the number of bees required to explore the solution space accurately should be increased to ensure satisfactory performance. This results from the modifications made in the loyalty assessment and stepwise filtering assignment steps. Once the number of bees is sufficient, then a sufficient amount of information will be available to measure the loyalty degree. Once the truly loyal bees are identified, this will affect the recruiter selection step positively, in the sense that the uncommitted bees can better select their recruiters, and consequently lead the algorithm to the preservation of more accurate solutions.

Similarly, the modifications made to the conventional BCO were shown to be effective in application. FS-*w*BCO experiments were carried out using the SVM classifier and benchmark datasets obtained from the UCI machine learning repository. FS-*w*BCO could improve conventional BCO-based feature selection, and gain superiorities over BCOFS. It also showed superiorities over other swarm and evolutionary-based algorithms, but had some inferiorities to

its competitors (also demonstrated in the Wilcoxon signed-ranks tests). This could result from dataset division as a hold-out strategy was used. FS-*w*BCO and BPSO were mainly similar with superiorities of FS-*w*BCO shown in a few datasets.

As part of our future work, we plan to further investigate the proposed algorithms with more functions. The FS-*w*BCO algorithm will be executed with other classifiers such as decision trees, ANNs, *k*-nearest neighbors, etc. and experiments will be conducted using alternative methodologies such as leave one out cross validation or *n*-fold cross validation procedures. Also, *w*BCO will be applied to regression problems, and will be improved to be applicable for continuous problems. The local weighting procedure will be modified to measure the correlation between the features and the dependent variables.

References

[1] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 1995, pp. 1942-1948.

[2] M. Dorigo, G.D. Caro, and L. M. Gambardella, "Ant Algorithms for Discrete Optimization," *Artificial Life*, vol. 5, no. 2, pp. 137-172, 1999.

[3] M. Nikolic and D. Teodorovic, "Empirical study of the Bee Colony Optimization (BCO) algorithm," *Expert Systems with Applications*, vol. 40, pp. 4609-4620, 2013.

[4] D. Teodorovic, P. Lucic, G. Markovic, and M. Dell' Orco, "Bee Colony Optimization: Principles and Applications," in *Neural Network Applications in Electrical Engineering, NEUREL 2006, 8th Seminar on*, 2006, pp. 151-156.

[5] P. Dziwiński, Ł. Bartczuk, and J.T. Starczewski, "Fully Controllable Ant Colony System for Text Data Clustering," in *Lecture Notes in Computer Science*., 2012, vol. 7269, pp. 199-205.

[6] R. Forsati, A. Moayedikia, R. Jensen, M. Shamsfard, and M.R. Meybodi, "Enriched ant colony optimization and its application in feature selection," *Neurocomputing*, vol. 142, pp. 354-371, 2014.

[7] R. Forsati, A. Moayedikia, and A. Keikhah, "A Novel Approach for Feature Selection based on the Bee Colony Optimization," *International Journal of Computer Applications*, vol. 43, no. 8, pp. 30-34, 2012.

[8] A. Unler and A. Murat., "A discrete particle swarm optimization method for feature selection in binary classification problems," *European Journal of Operational Research*, vol. 206, pp. 528–539, 2010.

[9] F. Ghareh Mohammadi and M. Saniee Abadeh, "Image steganalysis using a bee colony based feature selection algorithm," *Engineering Applications of Artificial Intelligence*, vol. 31, pp. 35-43, 2014.

[10] B. Li, Y. Li, and L. Gong, "Protein secondary structure optimization using an improved artificial bee colony algorithm based on AB off-lattice model," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 70-79, 2014.

[11] R. Forsati, A. Keikhah, and M. Shamsfard, "An improved bee colony optimization algorithm with an application to document clustering," *Neurocomputing*, 2015.

[12] F. Kang, J.J. Li, and Z.Y. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization for numerical functions," *Information Sciences*, vol. 12, pp. 3508 - 3531, 2011.

[13] D. Chyzhyk, A. Savio, and M. Grana, "Evolutionary ELM wrapper feature selection for Alzheimer's disease CAD on anatomical brain MRI," *Neurocomputing*, vol. 128, pp. 73-80, 2014.

[14] Q. Shen and R. Jensen, "Selecting informative features with fuzzy-rough sets and its application for complex systems monitoring," *Pattern Recognition*, vol. 37, no. 7, pp. 1351–1363, 2004.

[15] R. Jensen and Q. Shen, "Fuzzy–rough attribute reduction with application to web categorization.," *Fuzzy Sets and Systems.*, vol. 141, no. 3, pp. 469–485, 2004.

[16] S.F. Da Silva, M.X. Ribeiro, J.D.E.S. Batista Neto, C. Traina-Jr, and A.J.M. Traina, "Improving the ranking quality of medical image retrieval using a genetic feature selection method," *Decision Support Systems*, vol. 51, pp. 810-820, 2011.

[17] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*.: Springer Science & Business Media, 1998.

[18] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, no. 1-4, pp. 61-85, 2009.

[19] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications.," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21-57, 2014.

[20] R. Kumar, A. Sadu, R. Kumar, and S.K. Panda, "A novel multi-objective directed bee colony optimization algorithm for multi-objective emission constrained economic power dispatch," *Electrical Power and Energy Systems*, vol. 43, pp. 1241-1250, 2012.

[21] P. Lu, J. Zhou, H. Zhang, R. Zhang, and C. Wang, "Chaotic differential bee colony optimization algorithm for dynamic economic dispatch problem with valve-point effects," *Electrical Power and Energy Systems*, vol. 62, pp. 130-143, 2014.

[22] Y.M. Huang and J.C. Lin, "A new bee colony optimization algorithm with idle-time-based filtering scheme for open shop-scheduling problems," *Expert Systems with Applications*, vol. 38, pp. 5438-5447, 2011.

[23] R. Kumar, "Directed Bee Colony Optimization Algorithm," *Swarm and Evolutionary Computation*, vol. 17, pp. 60–73, 2014.

[24] M. Alzaqebah and S. Abdullah, "Hybrid bee colony optimization for examination timetabling problems," *Computers & Operations Research*, vol. 54, pp. 142-154, 2014.

[25] D. Karaboga and B. Akay, "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems," *Applied Soft Computing*, vol. 11, pp. 3021 - 3031,

2011.

[26] W. Gao and S. Liu, "Improved artificial bee colony algorithm for global optimization," *Information Processing Letters*, vol. 111, pp. 871-882, 2011.

[27] G. Li, P. Niu, and X. Xiao, "Development and investigation of efficient artificial bee colony algorithm for numerical fnction optimization," *Applied Soft Computing*, vol. 12, pp. 320-332, 2012.

[28] R. Akbari, R. Hedayatzadeh, K. Ziarati, and B. Hassanizadeh, "A multi-objective artificial bee colony algorithm," *Swarm and Evolutionary Computation*, vol. 2, pp. 39-52, 2012.

[29] N. Imanian, M.H. Shiri, and P. Moradi, "Velocity based artificial bee colony algorithm for high dimensional continuous optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 36, pp. 148-163, 2014.

[30] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, pp. 5682-5687, 2010.

[31] M.H. Kashan, N. Nahavandi, and A.H. Kashan, "DisABC: A new artificial bee colony algorithm for binary optimization," *Applied Soft Computing*, vol. 12, no. 1, pp. 342-352, 2012.

[32] R. Forsati, A. Moayedikia, and B. Safarkhani, "Heuristic Approach to Solve Feature Selection Problem," in *Lecture Notes in Computer Science*. Springer, Verlag Berlin Heidelberg, 2011, pp. 707-717.

[33] M.D. Monirul Kabir, M. Shahjahan, and K. Murase, "A New Local Search based Hybrid Genetic Algorithm for Feature Selection," *Neurocomputing*, vol. 74, pp. 2914–2928, 2011.

[34] J. Cohen., "A coefficient of agreement for nominal scales.," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[35] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information Fusion*, vol. 6, no. 1, pp. 83-98, 2005.

[36] A. Unler, A. Murat, and R.B. Chinnam, "mr^{2}PSO: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification," *Information Sciences*, vol. 181, pp. 4625–4641, 2011.

[37] J. Huang, Y. Cai, and X. Xu, "A hybrid genetic algorithm for feature selection wrapper based on mutual information.," *Pattern Recognition Letters.*, vol. 28, pp. 1825–1844, 2007.

[38] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of*

Machine Learning Research, vol. 7, pp. 1-30, 2006.

[39] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 80-83, 1945.